Morphological parsing on Tolkappiyam's perspective

Rajendran Sankaravelayuthan
rajushush@gmail.com

Tolkappyiam is the earliest grammar of Tamil which gives the full grammatical description of Tamil. Though it reflects on the early grammatical perspectives and has its foundation on the data belonging to the early old Tamil, its nature of grammatical description holds good for the present day Tamil data too and can be made use as a strong formalism in building any natural language processing systems. This paper views Tolkappiyam in this perspective and explains how the grammatical formalism of Tolkappiyam can be made use of for parsing Tamil words by a machine.

Tolkappiyam and the three models of morphological description

Three models of morphological description have been proposed for the analysis of words: Item and Arrangement (IA) (Hocket, 1958), Item and Process (IP) (Hocket, 1958) and Word and Paradigm (WP) (Robins, 1959). In the IA model of morphological description, words are seen as linear sequences of morphs. For instance, the word *vandtaan* 'he came' is split into three independent segments: *vaa-* 'come', *-ndt-* 'past tense' and *–aan* 'he'. The IP model of analysis views morphology as a set of derivational process, acting on word-forms, producing new word-forms. The theory works on the conception that word-forms constitute paired features of structure and function, or rather form and meaning, and the elements undergo a variety of modification, through transformations, to result in newer forms. For example *vandtaan* is derived form the analytical structure *vaa* + Past Tense + Third Person Masculine Singular which get realized as *vandtaan* by the process of formation of words. The WP model takes the 'word' within a paradigm of variables, as its basic unit. For example, as per WP model *vandteen*, *varukiReen, varuveen, vandtaay, varukiRaaya, vandtaan, varukiRaan, varuvaan,* etc as paradigm variables of the word *vaa* 'come'.

Tolkappiyam's morphological description in the two chapters *ezhuttu* and *col* are the amalgamation of the three models. The grammatical description of Tolkappiyam takes care of the arrangement in which morphs occur and concatenate to form words, the process by which words are formed and the natures of words to be realized in various paradigms. Thus Tokappiyam's approach to word analysis can be implemented on any type of morphological processing through machine.

Finites state Automata and Tolkappiyam

One can easily model such a system with a finite-state machine (finite-state automaton or finite-state transition network) of the kind familiar from formal language theory. More formally, a finite automation is defined by the following five parameters:

Q: a finite set of N states q0, q1...,qN

Σ: a finite input alphabet of symbols

q0: the start state

F: the set of final states, $F \subseteq Q$

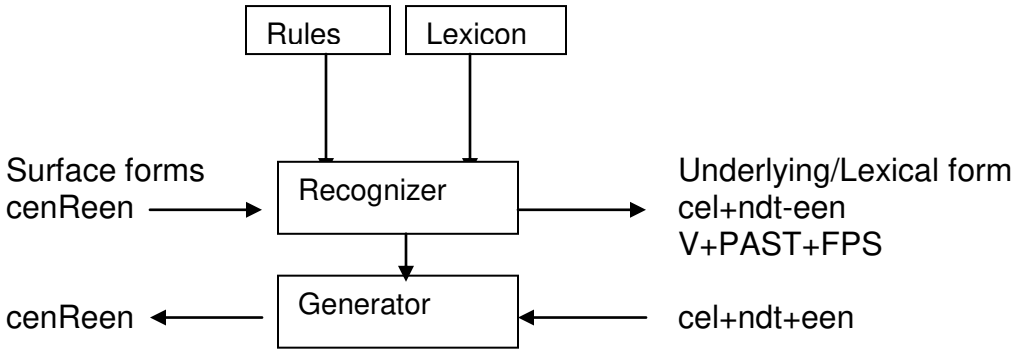
δ (q,i): the transition function or transition matrix between states. Given a state q $\in$ Q and an input symbol i $\in \Sigma$, δ (q,i) returns a new state q' $\in$ Q. δ is thus a relation from $Q \times \Sigma$ to Q;

A set of rewrite rules, or grammar, can be thought of not only as an intentional definition of the units of a language, but also as the description of an abstract machine which is able to accept (or generate) just those strings which belong to the language. These machines are called (abstract) automata, and the complexity of their potential behaviour is directly determined by the internal complexity of the rules of the grammar to which they correspond. All such machines may be characterized in terms of a set of states, an input device which can access one input symbol at a time, and control unit which can examine and read input and cause the machine to shift from one sate to another. In addition, such machines may have memory, which can be accessed by the control unit for storing and testing symbols. The computational power of the machine is determined essentially by the complexity of its memory and of the operations that can be performed on the memory. The simplest of these automata are called finite state machines or finite state automata (FSA), and the languages they can accept are regular languages.

The finite state automata checks words character by character and moves to the last character and either okays it as a possible form or reject it. The finite state transducer can take care of the change, deletion or addition of the characters so as to arrive at the surface realization of the word in question. Tolkappiyam by sandhi rules or morphonemic rules takes of the change, addition or deletion that occur when morphemes are concatenated. Thus Tolkappiyam gives a viable formalism which very well suit the implementation of morphological parsing.

Morphological processers and Tolkappiyam

The two level morphology of Kosekenniemi has been a widely implemented one, particularly in European languages. PCKimmo can do two functions as a morphological processor, recognition. In the case of the generator component of PCKimmo, it takes a lexical form of the foot as input. To this input rules recorded in the rules file are applied and the corresponding surface form is returned. It does not use the lexicon. The recognizer accepts as input a surface form, applies the rules, consults the lexicon and returns the corresponding lexical form and gloss string. What makes it easier for the user and important from the point of economy of model is the fact that the same set of rules can be used for both the functions of recognition and generation. The rules can be used bidirectionally (1990:8). If certain phonological, or of for that matter a morphological or orthographic, rule is written to refer to a transformation form the underlying structure to surface structure, they need not be rewritten for the reverse process. In reverse, it processes form the surface form to the underlying form. This model can be represented through the following diagram (Simons 1989).



(FPS=First person singular)

The two levels used in this model are linguistic descriptions that can handle any tow levels of abstractions, be it allophonic alternations, morphophonemic alternations or orthographic variations. PCkimmo translates these two level linguistic descriptions into finite state tables. To elaborate, the rules are represented as FINITE STATE TRANSDUCERS (FST) in the rule file and the morphotactic constraints are stated as FINITE STATE AUTOMATA (FSA) in the lexicon file. Both FST and FSA are finite state machines.

Tolkaappiyam naturally could be a chosen formalism which suits this model of morphological processing which includes analysis and generation. As in the case of two

level morphology, Tolkappiyam's approach to morphological processing also can take care of the two tire system of morphological processing. Its sandhi rules can be very easily converted into finite state transducers.

Take for example the inflection in *kaalam*. It can be attached to the demonstrative prefix *a-* as in *akkaalam* 'that period'. It can be suffixed with the different case suffixes like accusative *-ai*, dative *-ukku*, locative *-il*, locative-ablative *-illirundtu*, etc. and plural suffix *-kaL*, interrogative suffix *-aa*. All these morphological processings have been dealt by Tolkappiyam. The change of *kalam* into *kaalatt-* when case suffixes are added has been stated in Tolkappiyam, which can be very well taken care of by finite state transducers.

Not only any PC Kimo Model, any type of morphological modelling can be handled by making use of Tolkappiyam's approach to morphological processing. Many have worked on morphological processing of Tamil words using different kinds of formalism. For example, AMPLE is a morphological analyzer based on item and arrangement. It works by 'matching and filtering'. This base has let its designer build simpler, faster working and more general algorithms. The morphological processor Tagtamil by Ranganathan (1997) for Tamil word uses the linguistic theory of lexical phonology to organize the morphological data of the language. The basic insight of level-ordered morphology is that the derivational and inflectional processes of a language can be organized in a series of levels. Each level is associated with a set of phonological rules for which it defines the domain application. Vaishnavi's Tamil morphological processer as well as analyser is based on paradigm approach. She makes use of WP model for her processing. All these models do not escape the description given given by Tolkappiyam.

Conclusion

The aim of the present paper is not show or describe elaborately how to implement Tolkappiyam's morphological processing for building a morphological parser which works on machine, but to pin point that Tolkappiyam is a viable formalism which can be implemented for any kind of grammatical parsing which works on machine. Tolkappiyam does not leave anything unexplained, which is crucial for the preparation of morphological parser for Tami.

Reference:

Koskenniemi, Kimmo. 1983. Two-level Morphology: A General Computational Model for Word-Form Recognition and Production. Publication No. 11, Department of General Linguistics, Finland: University of Helsinki.

Rajendran, S. 1999. 'Spell and grammar checker for Tamil'. Paper read in 27th All India Conference of Dravidian Linguists held in ISDL, Thiruvananthapuram, 17th-19th, July, 1999.

Rajendran S, Arulmozi S, Ramesh Kumar S, & Viswanathan S. 2003. "Computational Morpohology of Verbal Complex", in B. Ramakrishna Reddy (ed.) Word Structure in Dravidian, Kuppam: Dravidian University, 376-398.

Ramaswamy, V. 2000. Morphological Generator for Tamil. Unpublished M.Phil. dissertation. University of Hyderabad.

Ranganathan, 1997. 'A Lexical Phonology Approach to Tamil Words by Computer', International Journal of Dravidian Linguistics 26:1, 57-70.

Vaishnavi Ramaswamy. 2000. A Morphological Generator for Tamil. M.Phil Dissertaion Subbmitted to University of Hyderabad.

Vaishnavi Ramaswamy. 2003. A Morphological Analyzer for Tamil. Ph.D Dissertaion Subbmitted to University of Hyderabad.

Vaishnavi, Ramaswamy. 2003. "Parsing in AMPLE, KIMMO & PERL: Nouns in Tamil", Paper presented in